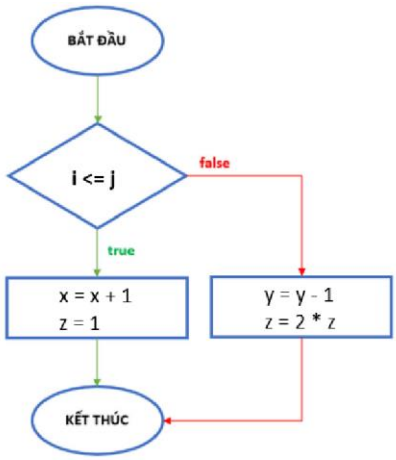
Nguyễn Đình Dương – 20225966

Assignment 1



i = 1, j = 2

```
1  # Laboratory Exercise 3, Home Assignment 1
2  .text
3  start:
4          li s1, 1
5          li s2, 2
6          slt t0, s2, s1      # set t0 = 1 if j < i else clear t0 = 0
7          bne t0, zero, else # t0 != 0 means t0 = 1, jump else
8  then:
9          addi  t1, t1, 1          # then part: x=x+1
10         addi  t3, zero, 1        # z=1
11         j endif               # skip "else" part
12 else:
13         addi  t2, t2, -1         # begin else part: y=y-1
14         add t3, t3, t3         # z=2*z
15 endif:
16
```

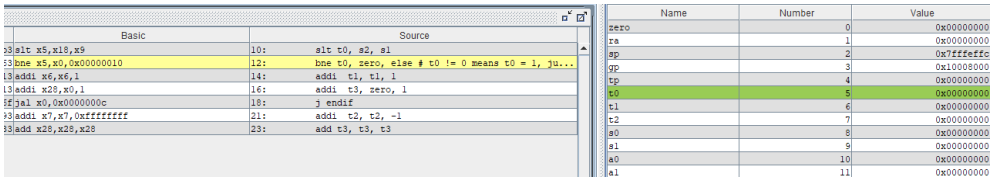
Kết quả chạy:

Text Segment

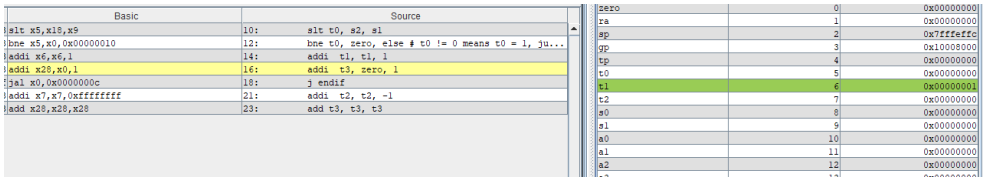
Bkpt	Address	Code	Basic	Source
	0x00400000	0x009922b3	slt x5,x18,x9	10: slt t0, s2, s1
	0x00400004	0x00029863	bne x5,x0,0x00000010	12: bne t0, zero, else # t0 != 0 means t0 = 1, ju...
	0x00400008	0x00130313	addi x6,x6,1	14: addi t1, t1, 1
	0x0040000c	0x00100e13	addi x28,x0,1	16: addi t3, zero, 1
	0x00400010	0x00c0006f	jal x0,0x0000000c	18: j endif
	0x00400014	0xfff38393	addi x7,x7,0xffffffff	21: addi t2, t2, -1
	0x00400018	0x01ce0e33	add x28,x28,x28	23: add t3, t3, t3

Kết quả:

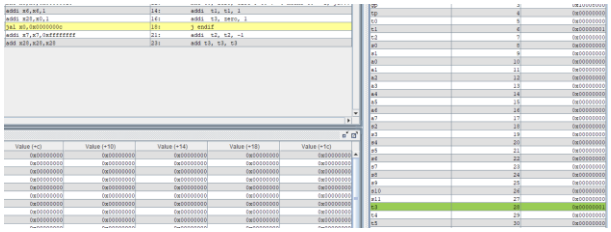
t0 = 0x00000000



t1=0x00000001



t3=0x00000001



Pc cứ mỗi step tăng lên 4 đơn vị

Giải thích:

1. Khởi tạo:

- li s1, 1 ⟶ s1 = 1
- li s2, 2 ⟶ s2 = 2

2. So sánh: slt t0, s2, s1 ⟶ t0 = 0 (vì 2 không nhỏ hơn 1) → Chương trình chạy sang nhánh then

3. Nhánh then:

- addi t1, t1, 1 ⟶ t1 = 1
- addi t3, zero, 1 ⟶ t3 = 1

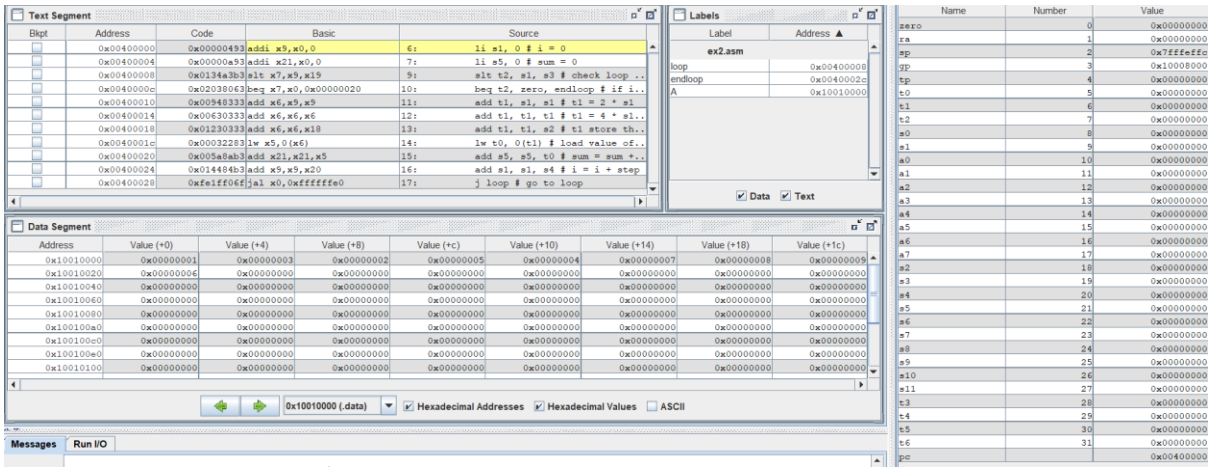
4. Kết thúc:

- Không thực hiện nhánh else.

Assignment 2

```
A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6
.text
        # TODO: Initialize s2, s3, s4 registers
        li s1, 0 # i = 0
        la s2, A # Địa chỉ của mảng
        li s3, 4 # Số phần tử của mảng
        li s4, 1 # Số bước nhảy
        li s5, 0 # sum = 0
loop:
        slt t2, s1, s3 # check loop condition i < n
        beq t2, zero, endloop # if i >= n then end loop
        add t1, s1, s1 # t1 = 2 * s1
        add t1, t1, t1 # t1 = 4 * s1 => t1 = 4*i
        add t1, t1, s2 # t1 store the address of A[i]
        lw t0, 0(t1) # load value of A[i] in t0
        add s5, s5, t0 # sum = sum + A[i]
        add s1, s1, s4 # i = i + step
        j loop # go to loop
endloop:
```

Kết quả chạy:



Giá trị các thanh ghi:

+ s2 sẽ được load địa chỉ của A: 0x10010004 sau đó giảm xuống 0x10010000

+ s3 sẽ được gán là 0x00000004 (số lần lặp), s4 là 0x00000001 (step)

+ qua mỗi bước lặp, giá trị t1 = 4 * s1, t0 gắn giá trị A[i] và s5 = s5 + t0 sau đấy quay lại kiểm tra điều kiện vòng lặp

+ s1 mỗi step sẽ nhảy 1 đơn vị, đến cuối cùng có gia trị: 0x00000004

+ pc cứ mỗi bước tăng 4 đơn vị

+ s5 cuối cùng có giá trị: 0x0000000b (tức là 11) (tổng 4 giá trị đầu là 11)

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000005
t1	6	0x1001000c
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000004
s4	20	0x00000001
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400040

Kiểm tra tính đúng đắn:

Tạo mảng mới với 4 phần tử: 2,2,2,2

Tổng phần từ sẽ là 8

```
.data
 A: .word 2, 2, 2, 2
.text
        # TODO: Initialize s2, s3, s4 registers
        li s1, 0 # i = 0
        la s2, A # Địa chỉ của mảng
        li s3, 4 # Số phần tử của mảng
        li s4, 1 # Số bước nhảy
        li s5, 0 # sum = 0
loop:
        slt t2, s1, s3 # check loop condition i < n
        beq t2, zero, endloop # if i >= n then end loop
        add t1, s1, s1 # t1 = 2 * s1
        add t1, t1, t1 # t1 = 4 * s1 => t1 = 4*i
        add t1, t1, s2 # t1 store the address of A[i]
        lw t0, 0(t1) # load value of A[i] in t0
        add s5, s5, t0 # sum = sum + A[i]
        add s1, s1, s4 # i = i + step
        j loop # go to loop
```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000002
t1	6	0x1001000c
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000004
s4	20	0x00000001
s5	21	0x00000008
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400040

Kết quả cuối cùng ra đúng như dự đoán

Assignment 3

Khởi tạo s2 = 1, s3 = 2

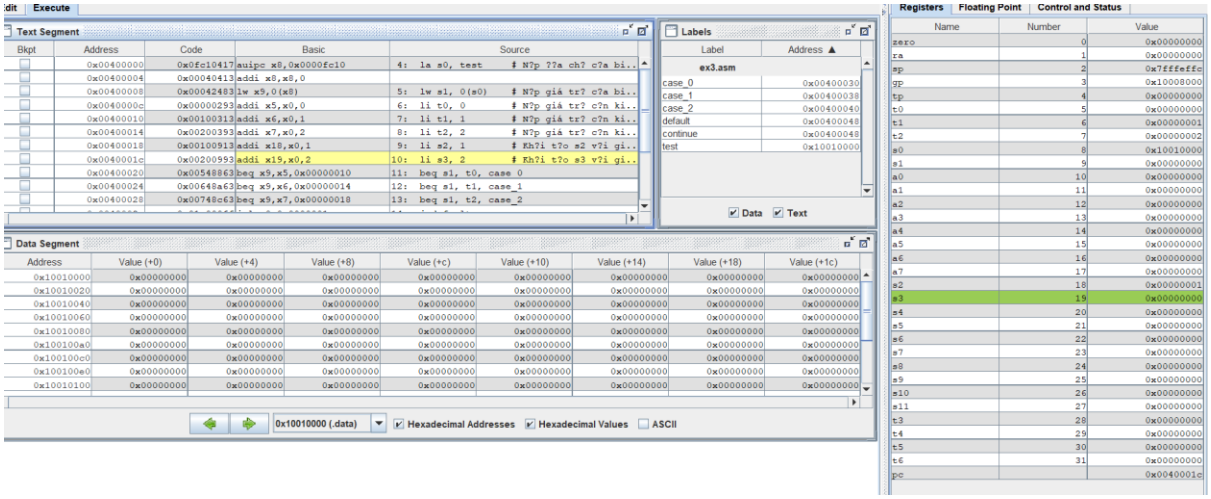
```
.data
 test: .word 0
.text
 la s0, test      # Nạp địa chỉ của biến test vào s0
 lw s1, 0(s0)     # Nạp giá trị của biến test vào s1
 li t0, 0         # Nạp giá trị cần kiểm tra
 li t1, 1         # Nạp giá trị cần kiểm tra
 li t2, 2         # Nạp giá trị cần kiểm tra
 li s2, 1         # Khởi tạo s2 với giá trị 1
 li s3, 2         # Khởi tạo s3 với giá trị 2
 beq s1, t0, case_0
 beq s1, t1, case_1
 beq s1, t2, case_2
 j default
case_0:
 addi s2, s2, 1  # a = a + 1
 j continue
case_1:
 sub s2, s2, t1  # a = a - 1
 j continue
case_2:
 add s3, s3, s3  # b = 2 * b
 j continue
default:
continue:
```

Các bước khởi tạo giá trị:

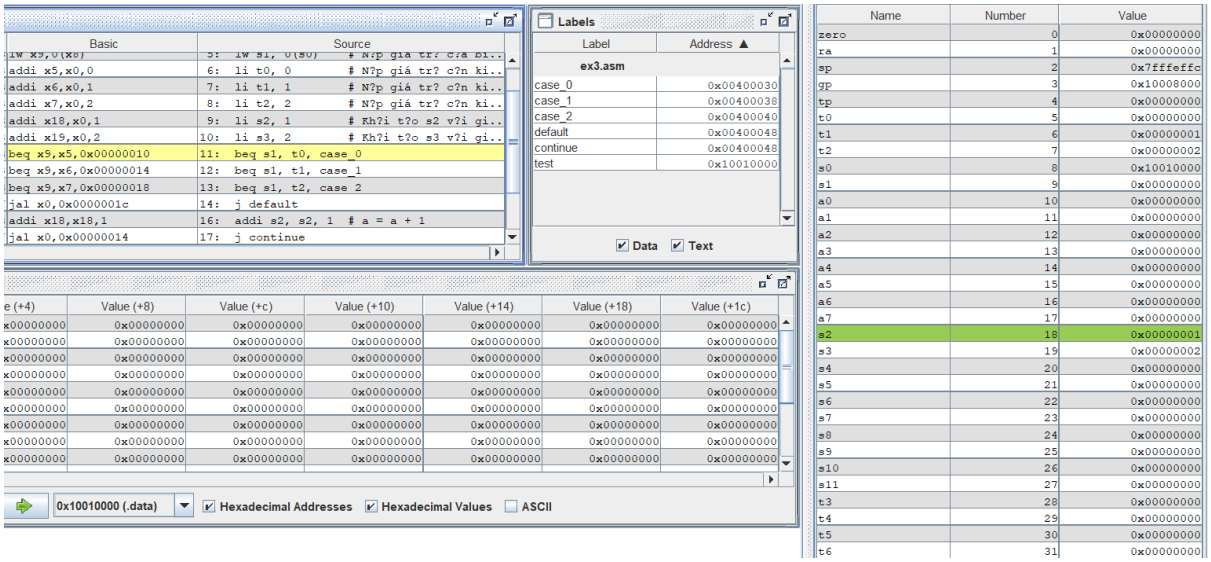
+ s0 gán địa chỉ của test ,s1 gán giá trị của biến test vào như hình

+ t0, t1, t2 gán giá trị của case

+ s2 gán bằng 1, s3 gán bằng 2



Do giá trị test = 0 nên sẽ nhảy vào case 0



S2 = s2 + 1 nên có giá trị : 0x00000002

Sau đó nhảy đến continue:

Kết quả cuối cùng:

(pc cứ mỗi bước tăng lên 1 đơn vị)

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000001
t2	7	0x00000002
s0	8	0x10010000
s1	9	0x00000000
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000002
s3	19	0x00000002
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040004c

Thử một vài trường hợp khác:

- Khi test bằng 2: thì giá trị của s3 = s3 * 2 : 0x00000004

```
.data
 test: .word 2
.text
 la s0, test      # Nạp địa chỉ của biến test vào s0
 lw s1, 0(s0)     # Nạp giá trị của biến test vào s1
 li t0, 0         # Nạp giá trị cần kiểm tra
 li t1, 1         # Nạp giá trị cần kiểm tra
 li t2, 2         # Nạp giá trị cần kiểm tra
 li s2, 1         # Khởi tạo s2 với giá trị 1
 li s3, 2         # Khởi tạo s3 với giá trị 2
 beq s1, t0, case_0
 beq s1, t1, case_1
 beq s1, t2, case_2
 j default
case_0:
 addi s2, s2, 1  # a = a + 1
 j continue
case_1:
 sub s2, s2, t1  # a = a - 1
 j continue
case_2:
 add s3, s3, s3  # b = 2 * b
 j continue
default:
continue:
```

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000001
t2	7	0x00000002
s0	8	0x10010000
s1	9	0x00000002
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000001
s3	19	0x00000004
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040004c

- Khi test là giá trị ngoài 0,1,2: sẽ nhảy vào case default và các giá trị sẽ giữ nguyên

```
.data
 test: .word 4
.text
 la s0, test      # Nạp địa chỉ của biến test vào s0
 lw s1, 0(s0)     # Nạp giá trị của biến test vào s1
 li t0, 0         # Nạp giá trị cần kiểm tra
 li t1, 1         # Nạp giá trị cần kiểm tra
 li t2, 2         # Nạp giá trị cần kiểm tra
 li s2, 1         # Khởi tạo s2 với giá trị 1
 li s3, 2         # Khởi tạo s3 với giá trị 2
 beq s1, t0, case_0
 beq s1, t1, case_1
 beq s1, t2, case_2
 j default
case_0:
 addi s2, s2, 1  # a = a + 1
 j continue
case_1:
 sub s2, s2, t1  # a = a - 1
 j continue
case_2:
 add s3, s3, s3  # b = 2 * b
 j continue
default:
continue:
```

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000001
t2	7	0x00000002
s0	8	0x10010000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000001
s3	19	0x00000002
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040004c

Assignment 4

a) I < j

```
.text
start:
    # Initialize i to s1
    li s1, 1
    # Initialize j to s2
    li s2, 2
    li t2, 2
    li t3, 2
    slt t0, s1, s2     # Set t0 = 1 if i < j, else 0
    bne t0, zero, else # If t0 != 0 (i < j), jump to else
then:
    addi t2, t2, -1    # Else part: y = y - 1
    add t3, t3, t3     # z = 2 * z
    j endif             # Skip else part
else:
    addi t1, t1, 1     # Then part: x = x + 1
    addi t3, zero, 1   # z = 1
endif:
```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000001
t1	6	0x00000001
t2	7	0x00000002
s0	8	0x00000000
s1	9	0x00000001
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000002
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000001
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400030

ở đây khởi tạo s1 = 1, s2 = 2 thì sẽ rơi vào trường hợp I > j thì sẽ thực thi lệnh x = x + 1, z = 1. Ta có thể thấy giá trị t1 = 0x00000001, t3 =0x00000001

b) I <= j

```
# Laboratory Exercise 3, Home Assignment 1
.text
start:
 # TODO:
        li s1, 1
        li s2, 1
         # Cách 1:
         # blt s2, s1, else # if j <= i then jump else
         # Cách 2:
        li t2, 10
         slt t0, s2, s1 # set t0 = 1 if j < i else clear t0 = 0
         beq t0, zero, else # If t0 == 0 (i >= j), jump to else
then:
         addi t1, t1, 1 # then part: x=x+1
         addi t3, zero, 1 # z=1
         j endif # skip "else" part
else:
         addi t2, t2, -1 # begin else part: y=y-1
         add t3, t3, t3 # z=2*z
endif:
```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000009
s0	8	0x00000000
s1	9	0x00000001
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000001
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040002c

Giá trị i =j nên sẽ nhảy sang else, t2 = t2 – 1 = 0x0000009

c) I + j <= 0

```
.text
start:
    li s1,1      # Initialize i to s1
    li s2,1# Initialize j to s2
    li t2,10
    add t0, s1, s2      # t0 = i + j
    slt t0, t0, zero    # Set t0 = 1 if (i + j) < 0, else 0
    beq t0, zero, else # If t0 == 0 (i + j > 0), jump to else
then:
    addi t1, t1, 1      # Then part: x = x + 1
    addi t3, zero, 1    # z = 1
    j endif              # Skip else part
else:
    addi t2, t2, -1     # Else part: y = y - 1
    add t3, t3, t3      # z = 2 * z
endif:
```

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000000
t2	7	0x00000009
s0	8	0x00000000
s1	9	0x00000001
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000001
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400030

Do I = 1, j = 1 nên sẽ rơi sang case else

t2 ban đầu là 10, sau khi chạy kết quả còn t2 = 0x00000009

d) i + j > m + n

```
.text
start:
        li s1,1
        li s2,1
        li s3,2
        li s4,2
        li t2,10
    # Assume m is in s3 and n is in s4
    add t0, s1, s2      # t0 = i + j
    add t1, s3, s4      # t1 = m + n
    slt t0, t1, t0      # Set t0 = 1 if (m + n) < (i + j), else 0
    beq t0, zero, else # If t0 == 0 (i + j <= m + n), jump to else
then:
    addi t1, t1, 1      # Then part: x = x + 1
    addi t3, zero, 1    # z = 1
    j endif              # Skip else part
else:
    addi t2, t2, -1     # Else part: y = y - 1
    add t3, t3, t3      # z = 2 * z
endif:
```

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000004
t2	7	0x00000009
s0	8	0x00000000
s1	9	0x00000001
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000001
s3	19	0x00000002
s4	20	0x00000002
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x0040003c

khởi tạo i = 1, j = 1, m =2, n = 2

do i + j < m + n nên sẽ rơi vào case else

nên giá trị t2 sẽ giảm đi một,

t2 = 0x00000009

Note: Các bài trên cứ mỗi bước pc sẽ tăng 4 đơn vị

Assignmment 5

a) I <= n

do I <= n nên sẽ chạy đúng n+1 lần do I bắt đầu từ 0.

```
.data
A: .word 1, 1, 1, 1, 1  # Mảng với 5 phần tử
.text
    # Khởi tạo các thanh ghi
    la s2, A            # Địa chỉ cơ sở của mảng A
    li s3, 4          # n = số phần tử của mảng A (5 phần tử)
    li s1, 0            # i = 0
    li s5, 0            # sum = 0
    li s4, 1            # step = 1
loop:
        bge  s3, s1, process
        blt s3, s1 endloop # if i >= n then end loop
process:
        add t1, s1, s1 # t1 = 2 * s1
        add t1, t1, t1 # t1 = 4 * s1 => t1 = 4*i
        add t1, t1, s2 # t1 store the address of A[i]
        lw  t0, 0(t1) # load value of A[i] in t0
        add s5, s5, t0 # sum = sum + A[i]
        add s1, s1, s4 # i = i + step
        j loop # go to loop
endloop:
```

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000001
t1	6	0x10010010
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000005
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000004
s4	20	0x00000001
s5	21	0x00000005
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400040

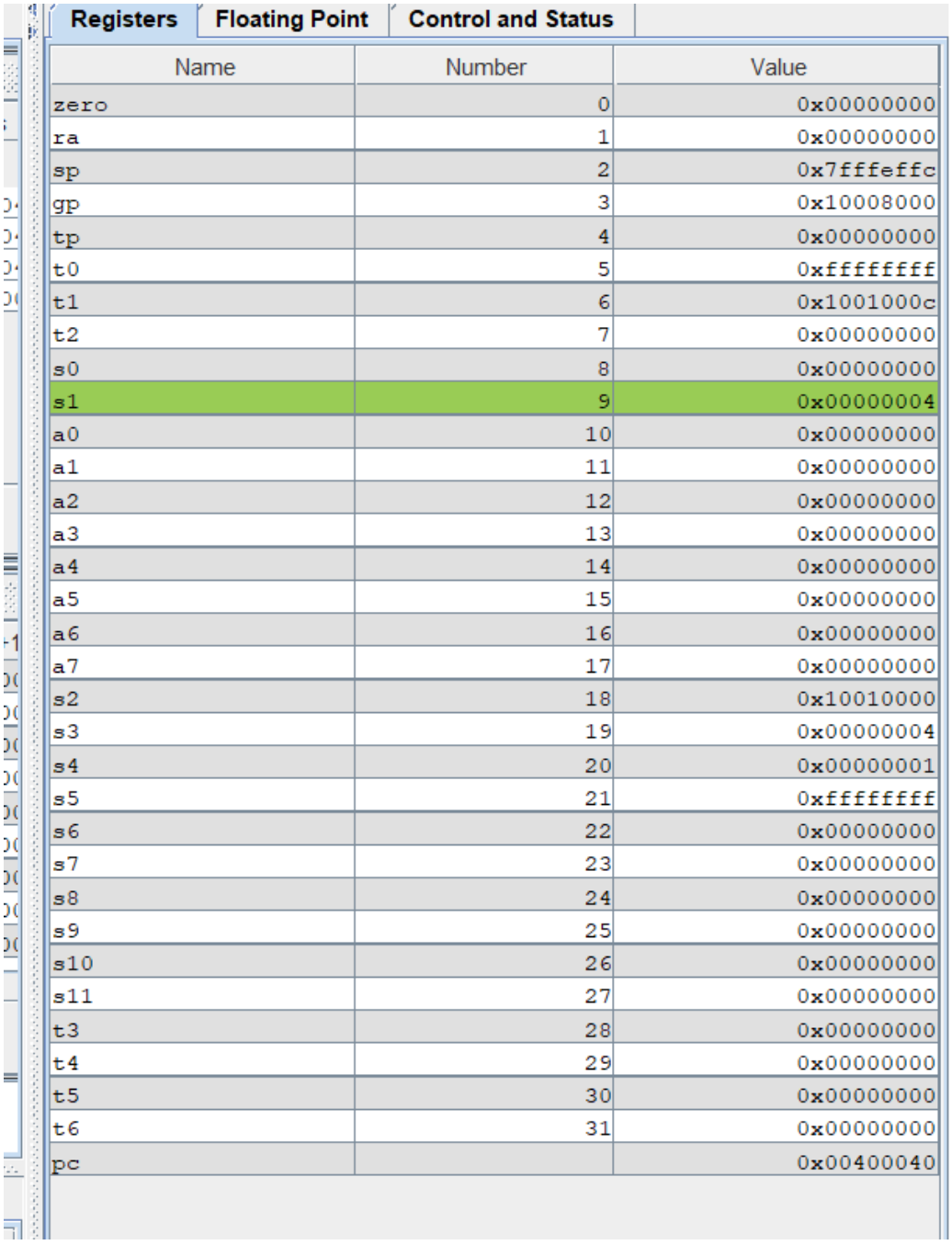
Giá trị của S5 sẽ bằng: 0x00000005

b) sum >= 0

vòng lặp sẽ dừng khi s5 = 0, bảng kết quả cho thấy s5: 0x00000000

ex2.asm | ex3.asm | ex1.asm | ex4-1.asm | ex4-2.asm | ex4-3.asm | ex4-4.asm | ex5-1.asm

```
.data
A: .word -1, -1, -1, -1, 1  # Mảng với 5 phần tử
.text
    # Khởi tạo các thanh ghi
    la s2, A            # Địa chỉ cơ sở của mảng A
    li s3, 4          # n = số phần tử của mảng A (5 phần tử)
    li s1, 0            # i = 0
    li s5, 4            # sum = 0
    li s4, 1            # step = 1
loop:
         blt   s5, zero, endloop
         bge s5, zero process # if i >= n then end loop
process:
         add t1, s1, s1 # t1 = 2 * s1
         add t1, t1, t1 # t1 = 4 * s1 => t1 = 4*i
         add t1, t1, s2 # t1 store the address of A[i]
         lw  t0, 0(t1) # load value of A[i] in t0
         add s5, s5, t0 # sum = sum + A[i]
         add s1, s1, s4 # i = i + step
         j loop # go to loop
endloop:
```



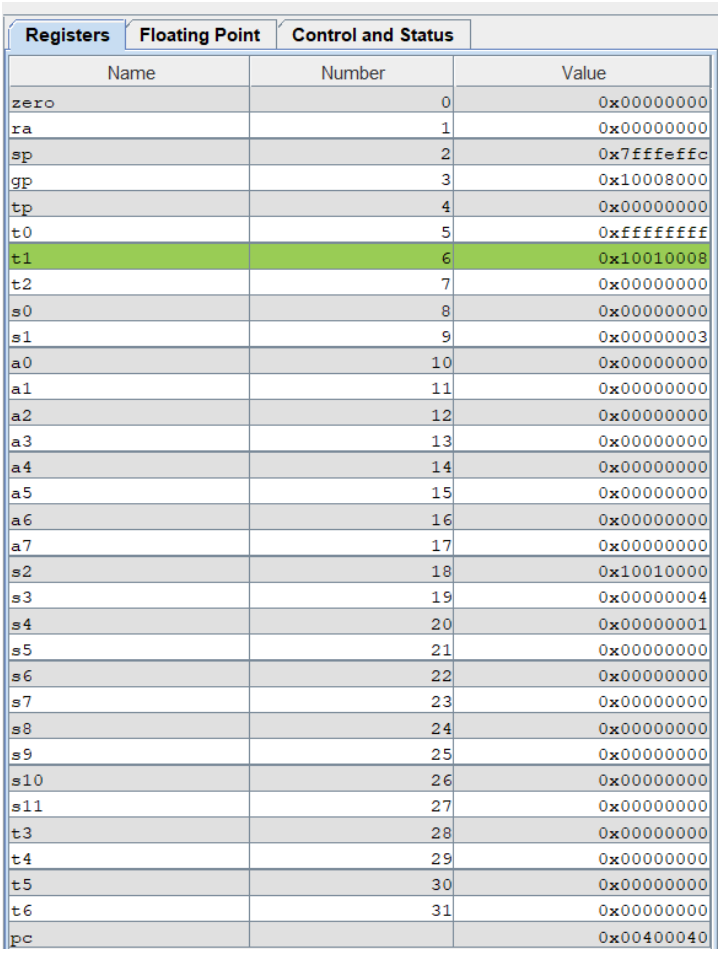
Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0xffffffff
t1	6	0x1001000c
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000004
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000004
s4	20	0x00000001
s5	21	0xffffffff
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400040

c) A[i] !=0

kết quả cho thấy vòng lặp sẽ dừng khi A[i] ==0

```
.data
A: .word -1, -1, -1, -1, 1  # Mảng với 5 phần tử
.text
    # Khởi tạo các thanh ghi
    la s2, A            # Địa chỉ cơ sở của mảng A
    li s3, 4          # n = số phần tử của mảng A (5 phần tử)
    li s1, 0            # i = 0
    li s5, 3            # sum = 0
    li s4, 1            # step = 1
loop:
        beq s5, zero, endloop
        bne  s5, zero,  process
process:
        add t1, s1, s1 # t1 = 2 * s1
        add t1, t1, t1 # t1 = 4 * s1 => t1 = 4*i
        add t1, t1, s2 # t1 store the address of A[i]
        lw  t0, 0(t1) # load value of A[i] in t0
        add s5, s5, t0 # sum = sum + A[i]
        add s1, s1, s4 # i = i + step
        j loop # go to loop
endloop:
```



Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0xffffffff
t1	6	0x10010008
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000003
a0	10	0x00000000
a1	11	0x00000000
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x10010000
s3	19	0x00000004
s4	20	0x00000001
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x00000000
t6	31	0x00000000
pc		0x00400040

Assigment 6:

```
.data
array:      .word   5, -6, 6, 72, -15, -80, -9   # Mảng chứa các số nguyên
array_size: .word   7                            # Số lượng phần tử trong mảng
```

```
max_abs:    .word   0                   # Biến lưu giá trị tuyệt đối lớn nhất
.text
.globl main
main:
    # Khởi tạo con trỏ mảng
    la      a0, array                   # a0 trỏ tới đầu mảng
    lw      t1, array_size              # t1 = số phần tử trong mảng
    li      t2, 0                       # t2 = chỉ số hiện tại (i = 0)
    li      t3, 0                       # t3 = giá trị tuyệt đối lớn nhất (max_abs)
loop:
    bge     t2, t1, end_loop            # Nếu i >= array_size thì kết thúc vòng lặp
    lw      t4, 0(a0)                   # Tải phần tử hiện tại vào t4
    # Tính giá trị tuyệt đối
    blt     t4, zero, neg_case          # Nếu t4 < 0, đi tới neg_case
    add     t5, zero, t4                # T5 = t4 nếu t4 >= 0
    j       check_max                   # Chuyển tới check_max
neg_case:
    sub     t5, zero, t4                # T5 = -t4 (tính giá trị tuyệt đối)
check_max:
    bgt     t5, t3, update_max          # Nếu giá trị tuyệt đối lớn hơn max_abs thì cập
nhật
    j       next_element                # Chuyển sang phần tử tiếp theo
update_max:
    add     t3, zero, t5                # Cập nhật max_abs với giá trị tuyệt đối lớn nhất
next_element:
    addi    a0, a0, 4                   # Di chuyển đến phần tử tiếp theo (4 bytes cho
int)
    addi    t2, t2, 1                   # Tăng chỉ số i
    j       loop                        # Quay lại vòng lặp
end_loop:
    la      a1, max_abs                 # Tải địa chỉ của max_abs vào a1
    sw      t3, 0(a1)                   # Lưu giá trị tuyệt đối lớn nhất vào biến max_abs
    # Kết thúc chương trình
    li      a7, 10                      # Mã syscall để kết thúc
    ecall
```

Registers | Floating Point | Control and Status

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
sp	2	0x7fffeffc
gp	3	0x10008000
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000007
t2	7	0x00000007
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x1001001c
a1	11	0x10010020
a2	12	0x00000000
a3	13	0x00000000
a4	14	0x00000000
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x0000000a
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x00000000
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x00000000
s8	24	0x00000000
s9	25	0x00000000
s10	26	0x00000000
s11	27	0x00000000
t3	28	0x00000050
t4	29	0xfffffff7
t5	30	0x00000009
t6	31	0x00000000
pc		0x0040005c